GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

An Inexpensive Firewall for the Home Network

By Nathan King
May 15, 2002
GSEC Practical Assignment v1.4, Option 2

Summary/Abstract

I decided to purchase high-speed access through my cable company to help me with my consulting, web design and hosting business. Of course, I needed a firewall but affective firewalls provided by most vendors can be expensive. Other firewalls available on the less expensive home cable/DSL routers are not fully functional and can be limiting. I needed a firewall that I could modify as my needs changed and one that I could afford. I decided to use a spare Intel PC and build my own firewall using a Unix operating system. This provided my internal network with Internet access and also provided network with a robust firewall.

Within I will show you what vulnerabilities I showed to the Internet before the firewall. Provide a basic description of a firewall. The steps I took to create my firewall, which included hardening of the firewall server. A detailed explanation of the important configuration files. And finally a scan after I put my firewall into place.

Before my Firewall

Below is an nmap scan without my firewall turned on. It shows some of the windows protocols leaking through (Port 139 and 135) from my Windows 98 PC as well as a few other interesting ports such as the telnet, ftp, pop and http services that was available on my gateway server.

```
nmap -sS a.b.c.d

Starting nmap V. 2.54BETA6 ( www.insecure.org/nmap/ )
Interesting ports on ip-a-b-c-d.myhome.net (a.b.c.d):
(The 1524 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
80/tcp     filtered    http
110/tcp    open        pop-3
111/tcp    filtered    sunrpc
119/tcp    filtered    nntp
135/tcp    filtered    loc-srv
139/tcp    filtered    netbios-ssn

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds
```


© SANS Institute 2000 - 2005 Author retains full rights.

An Inexpensive Firewall for the Home Network

© SANS Institute 2000 - 2005, Author retains full rights.

An Inexpensive Firewall for the Home Network

What is a firewall?

To begin with, a firewall is a device or devices applying methods of keeping the dangers of the Internet out of your home or business network. The firewall is where security policies determine what traffic comes in and what traffic goes out. Obviously, you will want to keep unwanted Internet traffic out of your internal networks. What often gets overlooked is that there is some internal traffic you do not want to go out of your network to the Internet. Some of this traffic could be viruses such as Nimbda and Code Red that infect Windows Internet Information Servers. There are also other viruses that send information about your computer back out to the Internet. Your security policies can also help to defeat these threats.

On a complex network you may want to allow external traffic in to your email and web servers. On simple home networks, that is often not needed. On my home network, I decided that I did not want to allow any traffic in from the Internet except for replies to my outgoing requests such as FTP, POP and HTTP. I also decided that I would only allow very specific information out of my network. Any traffic not specifically allowed will be denied by default. On my firewall I am using IP packet filtering with Network Address Translation (NAT) and Port Address Translation (PAT).

What I did for my firewall

I took a spare Intel 486dx4 100Mhz PC with 32MB ram and 1.2GB disk and equipped it with two network interface cards and loaded the OpenBSD UNIX operating system on it. I used an **ipf** to filter traffic and **ipnat** to convert my internal private IP addresses to a real world routable IP address. The IP address for my external interface card is being provided by my ISP through DHCP and I used a class C private IP address for my internal network.

Why OpenBSD?

Windows does not provide an affective firewall without costly third party programs such as Symantec Enterprise Firewall and I could not afford a hardware-based solution such as Cisco's PIX Firewall, therefore I decided to use a Unix system. As with Windows, Linux has it's share of security problems and I felt its popularity was more related to good marketing and I needed something that would provide me more functionality and security.

OpenBSD is a free, BSD based UNIX operating system, which is proactively secure and provides integrated cryptography. OpenBSD is probably one of the few, if not only, operating system that is secure by design, boasting "four years without a remote hole in the default install".[1]

© SANS Institute 2000 - 2005 Author retains full rights.

Installing OpenBSD

OpenBSD is available from http://www.openbsd.org. A bootable CD can be ordered, or you can download it via FTP. Detailed instructions for installing OpenBSD can be found at http://www.openbsd.org/faq/faq4.html. I downloaded a boot floppy image of OpenBSD, created my boot floppy and booted my soon to be firewall from the diskette. I followed most of the guidelines for installation as noted in the documents mentioned previously.

During the network setup, I let the network interface card that will be connected to the Internet use DHCP to get its network information. For the network interface card on the internal network, I assigned the IP address of 192.168.250.1 with a netmask of 255.255.255.0. I do not specify the default route or name server since this information will come from the ISP via DHCP. I used the entire hard drive for the installation, but for simplicity, I only created three partitions. I created a root partition about 116M in size, the required swap partition sized at 64MB and the remainder I left to the **/usr** partition. Since I already had my external interface connected to the cable modem and since I chose DHCP for that interface, I chose the option to FTP the files directly from one of the OpenBSD servers on the Internet. The only libraries that I installed were the base system (**base28.tgz**), the **bsd** kernel, and the default configuration files (**etc28.tgz**), all of the others are not necessary. Part of securing the firewall dictates to only install what is necessary.

After the installation completed and the PC booted off of the hard drive, I performed some housekeeping. Since I did not create separate partitions for the **/var**, **/home** and **/tmp** directories they will be located on the root (/) partition. The **/var**, **/home** and **/tmp** directory structures are where logs are created, user files are stored and temporary files are created. I moved these directories to the **/usr** partition which has the largest amount of space so we don't fill up the root (**/**) partition. I then created links back to their previous locations in the root so they will still be available to the services that require access to them. To perform this, I used the following commands:

```
FW#> mv /var /usr
FW#> ln -s /usr/var /var
FW#> mv /home /usr
FW#> ln -s /usr/home /home
FW#> mv /tmp /usr
FW#> ln -s /usr/tmp /tmp
```

[1] http://www.openbsd.org

© SANS Institute 2000 - 2005 Author retains full rights.

Next, I edited the **/etc/sysctl.conf** file and uncomment the following line by removing the leading '#' sign:

```
net.inet.ip.forwarding=1 # 1=Permit forwarding (routing) of packets
```

This turns the PC into a router. The router will not pass traffic from one internal host to another internal host. Traffic that does not belong to a host on the internal network will be routed through the firewall to the Internet.

Hardening OpenBSD

OpenBSD is very secure by its default install. However, another principle in network security states to disable any services which are not necessary. Tofurther secure the system, I first edited the **/etc/rc.conf** file and turn off all unnecessary services, see **Appendix D**. Since this system is a firewall, I turned off everything but **inetd**, **ipf**, and **nat**, which are all services necessary to run the firewall.

I then edited the **/etc/inetd.conf** file and deleted all of the lines except for three lines which start services on request such as **telnet**, **ftp** and **pop**. These services I further protect by using the TCP wrappers program **tcpd** as follows:

/etc/inetd.conf

```
#
# Internet server configuration database
#
ftp         stream tcp nowait root /usr/libexec/tcpd  ftpd -US
pop3     stream tcp nowait root /usr/libexec/tcpd  popper  -s  -T 30
telnet    stream tcp nowait root /usr/libexec/tcpd  telnetd –k
```

There are two other files associated with the **tcpd** wrapper. Those files are **hosts.allow** and **hosts.deny**. In the **hosts.deny,** see below, I specify the default rule of deny everything to everyone. This is based upon the principle of *deny by default*.

/etc/hosts.deny

```
all: all
```

In the **hosts.allow** file, see below, I allow all access to **192.268.250.5**, only allow **telnet**

© SANS Institute 2000 - 2005 Author retains full rights.

access to **192.168.250.4**, and **ftp** access to **192.168.250.8**. The only access allowed to my firewall is from within the network only and is protected by the **tcpd** wrapper as well as the IP packet filter you will see described later.

/etc/hosts.allow

```
ALL:            192.168.250.5
telnetd:        192.168.250.4
ftpd:           192.168.250.8
```

All requests, both denied and allowed, that go through **tcpd** are logged to **/var/log/authlog**. All unauthorized access is also logged to the monitor, which is also the system console.

This successfully completes my hardening of OpenBSD. Some more steps that could and probably should be taken would be to setup some host based intrusion detection and remove the telnet and ftp services.

Setting up NAT

Next, I setup the Network Address Translation (NAT) rules. These rules translate IP addresses on the internal network which are private IP addresses, see Appendix E, to the IP address assigned on the outside interface by the ISP via DHCP. See Appendix C for the complete **ipnat.rules** configuration file.

The following rule sets up a transparent application proxy service for FTP connections to the Internet.

```
#-------------------------------------------------------------
#  provide support to proxy ftp for the internal net
#-------------------------------------------------------------
map ep0 192.168.250.0/24 -> ep0/32  proxy port ftp ftp/tcp
```

The next rule provides NAT and PAT for all TCP and UDP packets leaving the system.

```
#-------------------------------------------------------------
#  NAT and PAT everything else
#-------------------------------------------------------------
map ep0 192.168.250.0/24 -> ep0/32 portmap tcp/udp 10000:20000
```

© SANS Institute 2000 - 2005, Author retains full rights.

The remaining rule applies to all other non-TCP/UDP requests such as ICMP and IPSEC, which do not use ports. This rule performs a one to one translation from the internal network IP address to the external Internet address.

```
map ep0 192.168.250.0/24 -> ep0/32
```

Setting up the IP Packet Filter

Ipf performs the packet filtering and provides the bulk of our firewall support. It is an IP packet filtering utility that will inspect IP packets and act upon them as defined in the **/etc/ipf.rules** file. See Appendix B. **Ipf** processes the rules in a top down order and will use the rule that lasted matched. Our first set of rules applies the, ***deny by default*** rule.

The first set of rules we define are to block everything incoming and outgoing from all interfaces and matches all traffic:

```
#------------------------------------------------------------
# Block in/out by default
#------------------------------------------------------------
block in from any to any
block out from any to any
```

Next we allow anything in/out of the loopback interface. The **quick** option shown here and later cause the rule checking to perform the action specified and stops if it matches this rule. In this rule set, if it matches, we pass the packet and the quit further processing of the ipfilter.rules.

```
#------------------------------------------------------------
# Pass in/out anything for loopback
#------------------------------------------------------------
pass in quick on lo0
pass out quick on lo0
```

Next we allow anything in and out on the inside interface. This simplifies the rules quite a bit. A better rule of practice would be to specifically define what we allow in and out on that interface, as we will be doing on the external interface. These rules look like:

© SANS Institute 2000 - 2005 Author retains full rights.

```
#---------------------------------------------------------------
# pass in/out anything by default for internal interface
#    this simplifies rules, but could be risky if the
#    box was ever compromised
#---------------------------------------------------------------
pass in quick on ep1
pass out quick on ep1
```

Next, I explicitly deny all NetBIOS traffic. I have the IP packet filter respond with the TCP RST command so the sending machine will immediately shutdown the connection instead of waiting and timing out. This also assures that none of the NetBIOS traffic leaks across the firewall from the inside to the outside.

```
#---------------------------------------------------------------
# block all netbios - sending back reset to keep the host
#   from waiting on a response
#---------------------------------------------------------------
block return-rst in quick proto tcp from any to any port = 137
block return-rst in quick proto tcp from any to any port = 138
block return-rst in quick proto tcp from any to any port = 139
```

Next, setup the rules for the **dhcp** client since we need to get our IP address from the ISP. The **keep state** seen here and later is a short cut which allows packets back in after we have initiated an outgoing session.

```
#---------------------------------------------------------------
# bootp client
#---------------------------------------------------------------
pass in  on ep0 proto tcp/udp from any port = bootps to any port = bootpc keep state
pass out on ep0 proto tcp/udp from any port = bootpc to any port = bootps keep state
```

Next we allow the ICMP protocol in and out

```
#---------------------------------------------------------------
# Allow the ICMP (PING) protocol
#---------------------------------------------------------------
pass out on ep0 proto icmp from any to any
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
pass in  on ep0 proto icmp from any to any
```

Finally we get down to what we are going to allow out to the Internet. These set of rules allow the ntp, domain (dns name resolving), http and https (ssl), smtp and pop3 (email), telnet and ftp to be processed to the Internet.

```
#---------------------------------------------------------------
# All other normal protocols
#---------------------------------------------------------------
pass out on ep0 proto tcp/udp from any to any port = domain keep state
pass out on ep0 proto tcp from any to any port = https keep state
pass out on ep0 proto tcp from any to any port = www keep state
pass out on ep0 proto tcp from any to any port = smtp keep state
pass out on ep0 proto tcp from any to any port = pop3 keep state
pass out on ep0 proto tcp from any to any port = telnet keep state
pass out on ep0 proto tcp from any to any port = ftp keep state
pass out on ep0 proto tcp from any to any port = ftp-data keep state
```

Finally, since I have a real job that pays the bills, I set some rules that allow me to create a VPN connection via IPSEC to the VPN server where I work. This allows me to access the network and network services there. Many of the low-end commercial firewalls do not support the IPSEC protocol and is another reason why I put my own firewall together. Here we also see that we can filter by IP address at the network level all the way down to the host. Here I am allowing the host located at a.b.c.d to pass traffic back in to our network. The **keep state** keywords only work for TCP and UDP traffic which is why we have to put in a rule allowing the traffic back in.

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

```
#-----------------------------------------------------------
# vpn - using IPSEC protocol and ISAKMP authentication
#-----------------------------------------------------------
pass out on ep0 proto udp from any to any port = isakmp keep state
pass in  on ep0 proto udp from any to any port = isakmp keep state

pass in  on ep0 proto esp from a.b.c.d/255.255.255.255 to any
pass out on ep0 proto esp from any to any
#
```

After Snapshot

I browsed out to www.grc.com and let them probe my system passing ALL tests successfully. You can see the results in Appendix F.

I also used **nmap** performing a stealth scan of my system from an outside source. Here are the results:

```
Unix> nmap -sS a.b.c.d

Starting nmap V. 2.54BETA6 ( www.insecure.org/nmap/ )
All 1534 scanned ports on ip-a-b-c-d.myhome.net (a.b.c.d) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 147 seconds
```

Mission accomplished.

The only cost was for the hardware that I already had sitting in the corner not being used. An effective firewall for the poor man's budget. It does not have to be a super powerful PC since you will probably not have a large number of hosts accessing the Internet.

I now have a firewall that successfully filters all Internet traffic coming in to my home network. The only allowed Internet bound traffic is for HTTP, POP, SMTP, DNS, FTP and IPSEC to my work VPN server. All other traffic will be rejected and will not cross the firewall. This reduces the exposure of any viruses such as Nimbda and Code Red, which might make it into my home network from spreading outside of my network. This also significantly decreases the ability of viruses that capture private information on any PC's on the home network from sending that information back to the Internet. The only

© SANS Institute 2000 - 2005 Author retains full rights.

possible risks now are viruses that propagate via SMTP and HTTP. These risks should be countered by using and keeping current, an anti-virus software on all internal hosts and appropriate hardening of web servers on my home network.

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

References

"IP Filter – TCP/IP Firewall/NAT software", 5 May 2002, http://coombs.anu.edu.au/ipfilter

"IP Filter Examples", 5 May 2002, http://coombs.anu.edu.au/ipfilter/examples.html

"Using OpenBSD 2.9 As A Firewall/Gateway for Home DSL or Cable", 7 May 2002, http://www.drones.com/obsd29-fw.html

"Introduction to Firewalls", 6 May 2002, http://csrc.nist.gov/publications/nistpubs/800-7/node155.html

"Firewall Components", 6 May 2002, http://csrc.nist.gov/publications/nistpubs/800-7/node156.html

"Packet Filtering", 6 May 2002, http://csrc.nist.gov/publications/nistpubs/800-7/node157.html

"Which Protocols to Filter", 6 May 2002, http://csrc.nist.gov/publications/nistpubs/800-7/node158.html

"OpenBSD", 9 May 2002, http://www.openbsd.org

"OpenBSD Project Goals", 9 May 2002, http://www.openbsd.org/goals.html

"OpenBSD Installation Guide", 5 May 2002, http://www.openbsd.org/faq/faq4.html

"OpenBSD Networking", 6 May 2002, http://www.openbsd.org/faq/faq6.html

"OpenBSD Security", 5 May 2002, http://www.openbsd.org/security.html

D. Brent Chapman and Elizabeth D. Zwicky, "Building Internet Firewalls", O'Reilly & Associates, Inc, ISBN: 1-56592-124-0

Gibson Research Corporation, http://www.grc.com

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix A – OpenBSD

OpenBSD is a FREE multi-platform BSD based Unix operating system. The focus of OpenBSD.org is to provide a Unix operating system, which is portable, proactively secure with integrated cryptography. It supports binary emulation of most programs from SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS and HP-UX. It also contains OpenSSH, which supports both SSH1 and SSH2.

More information regarding OpenBSD can be found at http://www.openbsd.org.

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix B – ipf.rules

```
#--------------------------------------------------------------
#  ipf Packet Filtering Rule Set
#
#  The basic premise of this firewall is to deny by default
#  then explicitly allow only services which we allow
#--------------------------------------------------------------

#--------------------------------------------------------------
# Block in/out by default
#--------------------------------------------------------------
block in from any to any
block out from any to any

#--------------------------------------------------------------
# Pass in/out anything for loopback
#--------------------------------------------------------------
pass in quick on lo0
pass out quick on lo0

#--------------------------------------------------------------
# pass in/out anything by default for internal interface
#    this simplifies rules, but could be risky if the
#    box was ever compromised
#--------------------------------------------------------------
pass in quick on ep1
pass out quick on ep1

#--------------------------------------------------------------
# block all netbios - sending back reset to keep the host
#   from waiting on a response
#--------------------------------------------------------------
block return-rst in quick proto tcp from any to any port = 137
block return-rst in quick proto tcp from any to any port = 138
block return-rst in quick proto tcp from any to any port = 139

#--------------------------------------------------------------
# bootp client
#--------------------------------------------------------------
pass out on ep0 proto tcp/udp from any to any port = bootps keep state
pass in  on ep0 proto tcp/udp from any to any port = bootpc keep state

#--------------------------------------------------------------
# Allow the ICMP (PING) protocol
#--------------------------------------------------------------
pass out on ep0 proto icmp from any to any
pass in  on ep0 proto icmp from any to any
```

© SANS Institute 2000 - 2005, Author retains full rights.

```
#--------------------------------------------------------------
# All other normal protocols
#--------------------------------------------------------------
pass out on ep0 proto tcp/udp from any to any port = ntp keep state
pass out on ep0 proto tcp/udp from any to any port = domain keep state
pass out on ep0 proto tcp from any to any port = whois keep state
pass out on ep0 proto tcp from any to any port = https keep state
pass out on ep0 proto tcp from any to any port = www keep state
pass out on ep0 proto tcp from any to any port = 911 keep state
pass out on ep0 proto tcp from any to any port = smtp keep state
pass out on ep0 proto tcp from any to any port = nntp keep state
pass out on ep0 proto tcp from any to any port = pop3 keep state
pass out on ep0 proto tcp from any to any port = telnet keep state
pass out on ep0 proto tcp from any to any port = ftp keep state
pass out on ep0 proto tcp from any to any port = ftp-data keep state

#--------------------------------------------------------------
# vpn - using IPSEC protocol and ISAKMP authentication
#--------------------------------------------------------------
pass out on ep0 proto udp from any to any port = isakmp keep state
pass in  on ep0 proto udp from any to any port = isakmp keep state

pass in  on ep0 proto esp from a.b.c.d/255.255.255.255 to any
pass out on ep0 proto esp from any to any
#
```

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix C – ipnat.rules

```
#---------------------------------------------------------------
#  provide support to proxy ftp for the internal net
#---------------------------------------------------------------
map ep0 192.168.250.0/24 -> ep0/32  proxy port ftp ftp/tcp
#---------------------------------------------------------------
#  NAT and PAT everything else
#---------------------------------------------------------------
map ep0 192.168.250.0/24 -> ep0/32 portmap tcp/udp 10000:20000
map ep0 192.168.250.0/24 -> ep0/32
```

© SANS Institute 2000 - 2005, Author retains full rights.

Appendix D – rc.conf

```
#!/bin/sh -
#
#       $OpenBSD: rc.conf,v 1.53 2000/10/16 14:57:13 aaron Exp $

# set these to "NO" to turn them off.  otherwise, they're used as flags
routed_flags=NO         # for normal use: "-q"
mrouted_flags=NO        # for normal use: "", if activated
                        # be sure to enable multicast_router below.
rarpd_flags=NO          # for normal use: "-a"
bootparamd_flags=NO     # for normal use: ""
rbootd_flags=NO         # for normal use: ""
sendmail_flags=NO       # for normal use: "-bd -q30m"
smtpfwdd_flags=NO       # for normal use: "", and no "-bd" above.
named_flags=NO                  # for normal use: ""
rdate_flags=NO          # for normal use: name of RFC868 timeserver
timed_flags=NO          # for normal use: ""
ntpdate_flags=NO        # for normal use: NTP server;
photurisd_flags=NO      # for normal use: ""
isakmpd_flags=NO        # for normal use: ""
mopd_flags=NO           # for normal use: "-a"
httpd_flags=NO          # for normal use: ""
apmd_flags=NO           # for normal use: ""
dhcpd_flags=NO                  # for normal use: "-q"

rtadvd_flags=NO         # for normal use: list of interfaces
                        # be sure to set net.inet6.ip6.forwarding=1

route6d_flags=NO        # for normal use: ""
                        # be sure to set net.inet6.ip6.forwarding=1

rtsold_flags=NO         # for normal use: interface
                        # be sure to set net.inet6.ip6.forwarding=0
                        # be sure to set net.inet6.ip6.accept_rtadv=1

# Set to NO if ftpd is running out of inetd
ftpd_flags=NO           # for non-inetd use: "-D"

# Set to NO if identd is running out of inetd
identd_flags=NO         # for non-inetd use: "-b -u nobody -elo"

# On some architectures, you must also disable
# console getty in /etc/ttys
xdm_flags=NO            # for normal use: ""

# For enabling console mouse support (i386 architecture only)
moused_flags=NO         # for ps/2 try: "-p /dev/psm0",  serial: "-p /dev/cua00"
```

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

```
# set the following to "YES" to turn them on
rwhod=NO
nfs_server=NO          # see sysctl.conf for nfs client configuration
lockd=NO
gated=NO
kerberos_server=NO     # kerberos server. run 'info kth-krb' for assistance.
kerberos_slave=NO      # kerberos slave server.
amd=NO
ipfilter=YES
ipnat=YES              # for "YES" ipfilter must also be "YES"
portmap=NO             # almost always needed
inetd=YES              # almost always needed
lpd=NO                 # printing daemons
check_quotas=NO        # NO may be desireable in some YP environments
sshd=NO                        # if YES, run sshd
ntpd=NO                        # run ntpd if it exists
afs=NO                 # mount and run afs

# Multicast routing configuration
# Please look at /etc/netstart for a detailed description if you
# change these
multicast_host=NO      # Route all multicast packets to a single interface
multicast_router=NO    # A multicast routing daemon will be run, e.g. mrouted

# miscellaneous other flags
# only used if the appropriate server is marked YES above
gated_flags=
ypserv_flags=                  # E.g. -1 for YP v1, -d for DNS etc
yppasswdd_flags=               # "-d /etc/yp" if passwd files are in /etc/yp
nfsd_flags="-tun 4"            # Crank the 4 for a busy NFS fileserver
amd_dir=/tmp_mnt               # AMD's mount directory
amd_master=/etc/amd/master     # AMD 'master' map
ipfilter_rules=/etc/ipf.rules  # Rules for IP packet filtering
ipnat_rules=/etc/ipnat.rules   # Rules for Network Address Translation
ipmon_flags=-Ds                # To disable logging, use ipmon_flags=NO
syslogd_flags=                 # add more flags, ie. "-u -a /chroot/dev/log"
named_user=named               # Named should not run as root unless neccesary
named_chroot=/var/named        # Where to chroot named if not empty
afs_mount_point=/afs           # Mountpoint for AFS
afs_device=/dev/xfs0           # Device used by afsd
afsd_flags=-z                  # Flags passed to afsd
shlib_dirs=                    # extra directories for ldconfig

local_rcconf="/etc/rc.conf.local"

[ -f ${local_rcconf} ] && . ${local_rcconf} # Do not edit this line
```

© SANS Institute 2000 – 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix E – Private IP Addresses

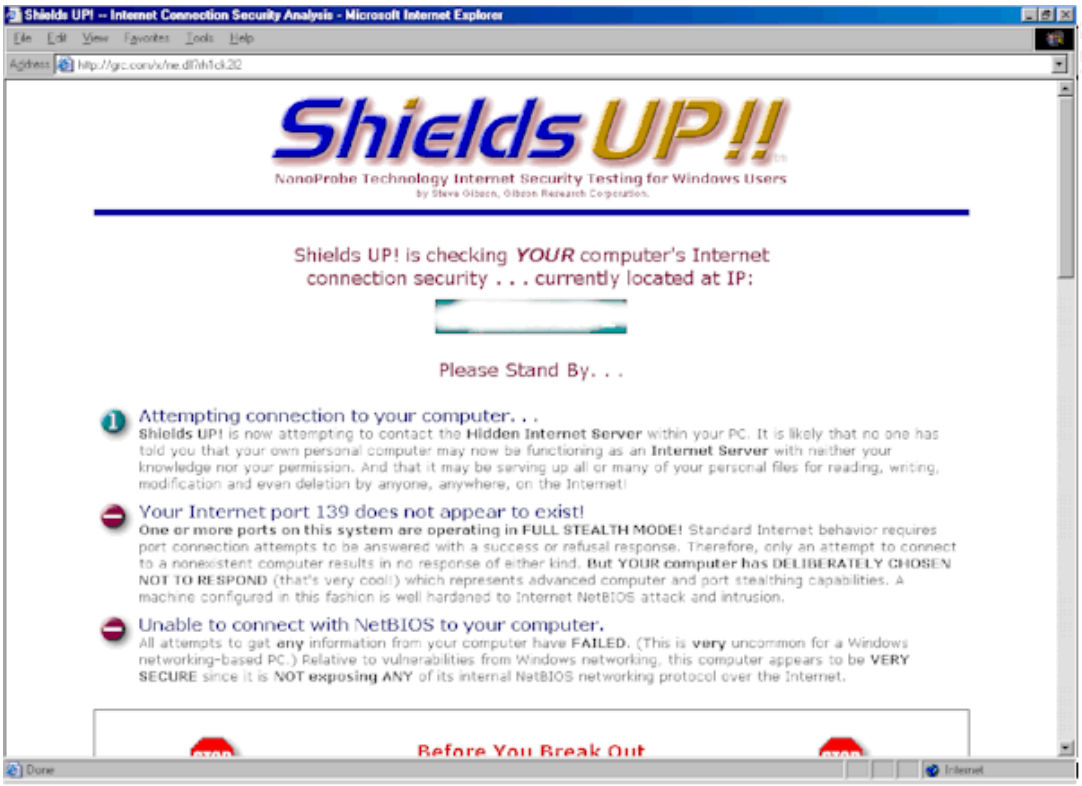
RFC1597 sets aside specific IP addresses that can be used by anyone for internal networks. The private IP Addresses set aside are:

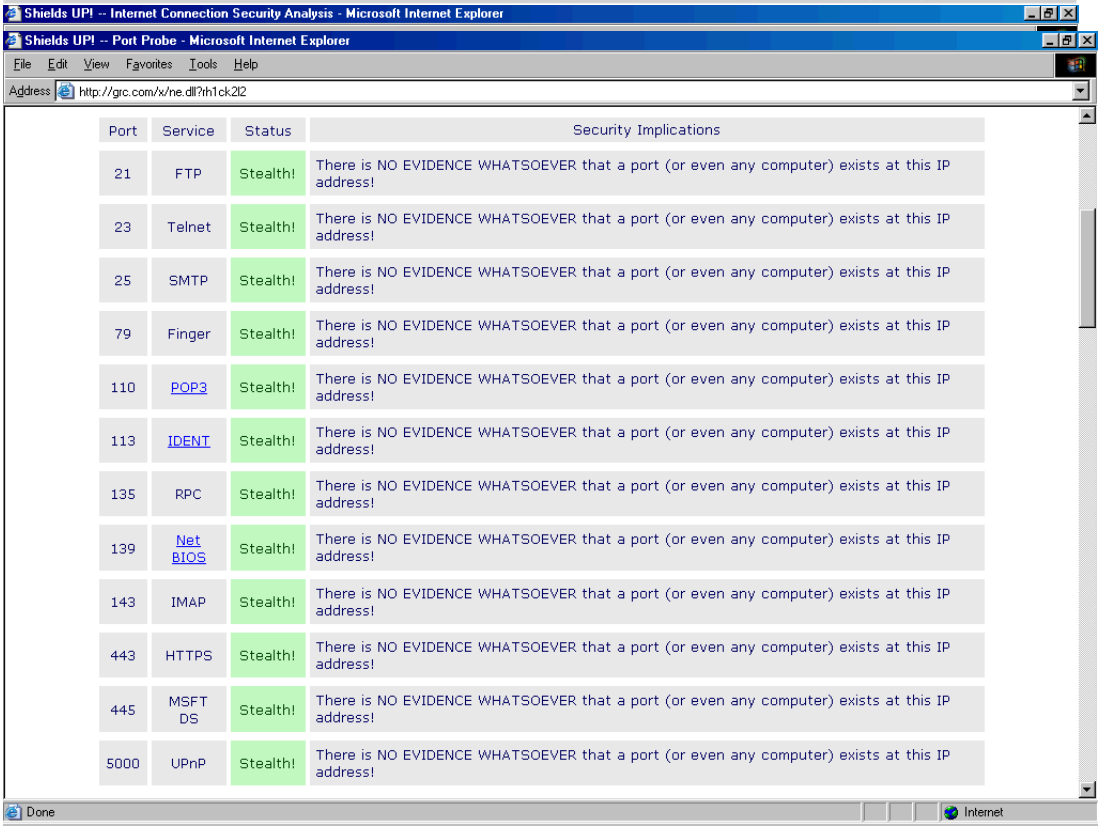
Class	Beginning IP	Ending IP
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.16.255.255
C	192.168.0.0	192.168.255.255

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix F – Results of WWW.GRC.COM test





Port	Service	Status	Security Implications
21	FTP	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
23	Telnet	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
25	SMTP	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
79	Finger	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
110	POP3	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
113	IDENT	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
135	RPC	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
139	Net BIOS	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
143	IMAP	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
443	HTTPS	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
445	MSFT DS	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
5000	UPnP	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!

© SANS Institute 2000 - 2005 Author retains full rights.